

Introduction to CVS

Josh Glover <colug@jmglov.net>
(2004/06/26)

*This file is licensed under the Creative Commons Attribution-ShareAlike License.
To view a copy of this license, visit:*

<http://creativecommons.org/licenses/by-sa/2.0/>

or send a letter to:

*Creative Commons
559 Nathan Abbott Way
Stanford, California 94305, USA.*

About

- CVS is the Concurrent Versions System
- Provides you with a way to record the revision history of any document
- You can select versions by revision number, date, tag, or branch
- Allows for concurrent editing of files

History (1/3)

- Started out as a bunch of shell scripts written by Dick Grune, posted to the newsgroup comp.sources.unix in the volume 6 release of July, 1986
- While no actual code from these shell scripts is present in the current version of CVS much of the CVS conflict resolution algorithms come from them

History (2/3)

- In April, 1989, Brian Berliner designed and coded CVS
- Jeff Polk later helped Brian with the design of the CVS module and vendor branch support

History (3/3)

- The future of CVS: Subversion
- The goal of the Subversion project is to build a version control system that is a compelling replacement for CVS in the open source community
- Written by CVS developers

Terminology

- Repository: central location where CVS (server) stores its files
- Sandbox: Local copy of a directory tree checked out from repository
- Module: Directory tree or trees that pertain to a single project

Commands (1/2)

- General form of CVS commands is:

```
cvcs {<global option>} <command> \  
    {<command option>} <name> ...
```

- Common global options:

```
-d <dir>          specify repository dir  
-e <editor>       use <editor> as your editor  
-f               ignore ~/.cvsrc  
-H [<cmd>]        display usage (for <cmd>)  
-n              do not change any files  
-q              be somewhat quiet  
-Q              be really quiet  
-r              make new working files read-only  
-t              generate trace messages  
-w              make new working files read-write (default)
```

Commands (2/2)

- Common command options:

-D <date>	use the latest version as of <date>
-f	ignore missing <date>s and <tag>s
-k <kflag>	alter default processing of keywords
-l	run only in local dir, with no recursion
-m <msg>	use <msg> as the log message
-n	do not run any tag program
-P	prune empty directories
-p	pipe filenames to STDOUT
-R	process dirs recursively (on by default)
-r <tag>	use <tag> instead of HEAD
-W	specify filenames to filter

Creating a Repository

- To create a repository:

```
mkdir ~/cvsroot
```

```
cvs -d ~/cvsroot init
```

- Global option `-d` specifies a repository directory
- The new repository will contain a single module: `CVSROOT`
- This module is used for CVS configuration files

Setting up Environment

- CVS uses two important environment variables:

`CVSROOT <dir>` repository root directory

`CVS_RSH <cmd>` remote shell command

- Let's point CVS at our newly created repository:

```
export CVSROOT=~ /cvsroot
```

- Let's use SSH as our remote shell command:

```
export CVS_RSH=ssh
```

Importing a Module

- To import a new module:

```
mkdir poetry
```

```
cd poetry
```

```
cvs import -m 'initial import' poetry \  
    JMGLOV START
```

- Command option `-m` is used to specify a log message on the command-line
- `poetry` is the module name
- `JMGLOV` is the vendor tag
- `START` is the release tag

Checking Out a Module

- To check out newly created module:
`cd`
`cv s co poetry`
- `co` is short form of checkout
- `poetry` is the module name
- Your `poetry` directory will now contain a CVS directory with the following files:
`Entries`
`Repository`
`Root`
- These three files are used by CVS to control each directory in your sandbox

CVS Sandbox Files: Entries

- Contains a line for each file or directory in the current directory
- Starts out with a single, empty directory entry:
: jmglov@delyana; cat CVS/Entries
D

CVS Sandbox Files: Repository

- Contains the location in the repository of this module
- In our case:
: jmglov@delyana; cat CVS/Repository
poetry

CVS Sandbox Files: Root

- Contains the location of the root of the CVS repository
- In our case:
: jmglov@delyana; cat CVS/Root
/home/jmglov/cvsroot/

Adding a File (1/3)

- Let's write a haiku:

```
cat >haiku.txt <<'EOF'
```

```
Brian Berliner;
```

```
The version control system,
```

```
CVS: safe code.
```

```
EOF
```

- `cvs add haiku.txt`

Adding a File (2/3)

- According to `CVS/Entries`, haiku has been added:

```
: jmglov@delyana; cat CVS/Entries  
/haiku.txt/0/Initial haiku.txt//
```

D

- But `cvstatus` tells a different tale:

```
: jmglov@delyana; cvs -Q status haiku.txt  
=====  
File: haiku.txt           Status: Locally Added
```

```
Working revision:      New file!
```

```
Repository revision:  No revision control file
```

Adding a File (3/3)

- Adding a new file to CVS is a two-step process
- First use `cv`s `add` to schedule the file for addition, then `cv`s `commit` to actually add it:
`cv`s `commit -m 'initial revision' \`
`haiku.txt`
- **Now, `cv`s `status` says what we would expect:**

```
: jmglov@delyana; cv
```

s -Q status haiku.txt

```
=====
```

```
File: haiku.txt          Status: Up-to-date
```

```
Working revision:      1.1      Sat Jun 26 02:17:06 2004
```

```
Repository revision:  1.1      /home/jmglov/cvsroot/poetry/haiku.txt,v
```

Ubiquitous Computing (1/4)

- CVS makes it easy to work on your files anytime, anywhere
- The muse strikes whilst I am reading in bed!
- I pick up my laptop, fire up the wireless LAN, and check out my `poetry` module:

```
cv$ -d \  
delyana: /home/jmglov/cvsroot \  
co poetry
```

Ubiquitous Computing (2/4)

- This time, a limerick:

```
cd poetry
```

```
cat >limerick.txt <<'EOF'
```

```
There was a young man named Jeff Polk
```

```
Many considered him "good folk"
```

```
He wrote module support for friend Brian,
```

```
And said to all those who were cryin'
```

```
"Don't fix it if it ain't broke!"
```

```
EOF
```

- `cvs add limerick.txt`
- `cvs commit -m 'initial revision' \
limerick.txt`

Ubiquitous Computing (3/4)

- My wife asks what I am doing, reads my haiku, and suggests a punctuation change:
Brian Berliner.
`*The* version control system,`
`CVS. Safe code.`
- I must commit my changes, but first I use `cvs diff` to see exactly what has changed

Ubiquitous Computing (4/4)

- : jmglov@laurana; cvs diff haiku.txt
Index: haiku.txt
=====
RCS file: /home/jmglov/cvsroot/poetry/haiku.txt,v
retrieving revision 1.1
diff -r1.1 haiku.txt
1,3c1,3
< Brian Berliner;
< The version control system,
< CVS: safe code.

> Brian Berliner.
> *The* version control system,
> CVS. Safe code.
- cvs commit -m 'changed punctuation' haiku.txt

Updating

- A few days later, when I return to my desktop box, I run `cvstatus` to see what to do:

```
: jmglov@delyana; cvs -Q status
```

```
=====  
File: haiku.txt           Status: Needs Patch
```

```
Working revision:      1.1   Sat Jun 26 02:17:06 2004
```

```
Repository revision:  1.2   /home/jmglov/cvsroot/poetry/haiku.txt,v
```

- A status of “Needs Patch” means that compatible changes have been made to the source file
- `: jmglov@delyana; cvs update -d`
`cvs update: Updating .`
`U haiku.txt`
`U limerick.txt`
- The “U” flags mean that the files have been updated

Merges (1/3)

- The “C” in CVS stands for concurrent, and CVS does in fact enable concurrent editing
- On my laptop, I add a title to my limerick:
`"Jeff Polk"`
`A Limerick by Josh Glover`
- I commit my changes like a good boy:
`cvsv commit -m 'added title' \
limerick.txt`

Merges (2/3)

- On my desktop, I add a copyright line:
Copyright (c) 2004, Josh Glover
- And then I remember that I had made some change to the limerick on my laptop!
- I turn to `cv`s status for guidance:

```
: jmglov@delyana; cvs -Q status limerick.txt
```

```
=====
```

```
File: limerick.txt          Status: Needs Merge
```

```
Working revision:      1.1 Sat Jun 26 10:45:22 2004
```

```
Repository revision:  1.2 /home/jmglov/cvsroot/poetry/limerick.txt,v
```

Merges (3/3)

- I grit my teeth and try to update:

```
: jmglov@delyana; cvs update limerick.txt
RCS file: /home/jmglov/cvsroot/poetry/limerick.txt,v
retrieving revision 1.1
retrieving revision 1.2
Merging differences between 1.1 and 1.2 into limerick.txt
M limerick.txt
```

- The “M” means that CVS has automatically merged my local version with the latest revision from the repository!

Conflicts (1/7)

- Emboldened by CVS's efforts on my part, I forge ahead
- I decide, since my name appears in the copyright notice, I can take it out of the title lines
- I commit:

```
cv$ commit \  
-m 'removed my name from title' \  
limerick.txt
```

Conflicts (2/7)

- Later, on my laptop, I decide there is no need to capitalise the “L” in “Limerick” in the title lines
- I make the change (without updating first!)
- I try to commit:

```
: jmglov@laurana; cvs commit \  
-m 'no need to capitalise "Limerick"' \  
limerick.txt  
cvs commit: Up-to-date check failed for `limerick.txt'  
cvs [commit aborted]: correct above errors first!
```
- Yowza! I really need to stop forgetting to update **before** I make changes!

Conflicts (3/7)

- I update from the repository:

```
: jmglov@laurana; cvs update limerick.txt
RCS file: /home/jmglov/cvsroot/poetry/limerick.txt,v
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into limerick.txt
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in limerick.txt
C limerick.txt
```

- The “C” means that CVS tried to automatically merge the two files, but conflicts arose
- This requires some action on my part, so I fire up my editor:
`vim limerick.txt`

Conflicts (4/7)

- The file looks like this:

```
"Jeff Polk"  
<<<<<<< limerick.txt  
A limerick by Josh Glover  
=====  
A Limerick  
>>>>>>> 1.3
```

```
There was a young man named Jeff Polk  
Many considered him "good folk"  
He wrote module support for friend Brian,  
And said to all those who were cryin'  
"Don't fix it if it ain't broke!"
```

```
Copyright (c) 2004, Josh Glover
```

Conflicts (5/7)

- Conflicts are always indicated like this:

```
<<<<<< local-file
local file text that doesn't
jive with repository version
=====
repository version text that
doesn't jive with local file
>>>>>> 1.3
```

Conflicts (6/7)

- This is an easy conflict to resolve
- I just choose the text that is closest to what I want (or easiest to change to what I want), in this case, the local file's text, and edit it:

```
A limerick
```

- Now, I get rid of the other four lines (the “<<<<” line, the “====” line, the repository version text, and the “>>>>” line)
- ***Now*** I can commit:

```
cv$ commit \  
    -m 'no need to capitalise "Limerick"' \  
    limerick.txt
```

Conflicts (7/7)

- The moral of this story is twofold:
- Conflicts do not often occur, and most are easily resolved
- ***Always*** update, or at the very least, run `cvstatus`, before editing a file!

Binary Files (1/3)

- Until now, we have been talking about CVS in the context of text files
- All of the utility of CVS is not lost when dealing with binary files, but they do have to be treated differently
- Imagine CVS trying to patch or merge a binary file!
- With binary files, merging must always be done manually
- CVS can still track revision history!

Binary Files (2/3)

- I want to add a new file to CVS: `haiku.wav`, which is me reading my haiku
- I copy the file into my sandbox dir:

```
cd ~/poetry/  
cp /tmp/haiku.wav .
```
- When I add it, however, I must instruct CVS to treat it differently:

```
cvs add -k 'b' haiku.wav  
cvs commit \  
    -m 'initial revision' haiku.wav
```
- The `-k 'b'` flag tells CVS to treat this file as binary

Binary Files (3/3)

- If a file is added with the 'b' flag, the flag becomes “sticky”, meaning that CVS will always treat it as a binary file
- This is well and good if you just have a few binary files, but what about a website, where you have tens or hundreds of image files, and you are adding more all the time? What if you forget the `-k 'b'` option?
- Luckily, CVS has config files that are global to a repository

The CVSROOT Module (1/2)

- When a new CVS repository is created with `cvs init`, a CVSROOT module is also created

- Let's check it out:

```
: jmglov@delyana; cd
: jmglov@delyana; cvs co CVSROOT
cvs checkout: Updating CVSROOT
U CVSROOT/checkoutlist
U CVSROOT/commitinfo
U CVSROOT/config
U CVSROOT/cvswrappers
U CVSROOT/editinfo
U CVSROOT/logininfo
U CVSROOT/modules
U CVSROOT/notify
U CVSROOT/rcsinfo
U CVSROOT/taginfo
U CVSROOT/verifymsg
```

The CVSROOT Module (2/2)

- Let's tell CVS about some files to always treat as binary:

```
: jmglov@delyana; cat >>cvswrappers <<'EOF'  
: ; *.bmp -k 'b'  
: ; *.gif -k 'b'  
: ; *.jpeg -k 'b'  
: ; *.jpg -k 'b'  
: ; EOF  
: jmglov@delyana; cvs commit -m 'added binary types' cvswrappers  
Checking in cvswrappers;  
/home/jmglov/cvsroot/CVSROOT/cvswrappers,v <-- cvswrappers  
new revision: 1.2; previous revision: 1.1  
done  
cvs commit: Rebuilding administrative file database
```

- Notice the last line: CVS has noticed that an administrative file has changed, and is rebuilding its “database”

Finis

- You now know enough about CVS to make you dangerous!
- Go forth, my sons (and daughters, as applicable), and use CVS to save thy revision history!
- A good next step is “The Cederqvist”:

<https://www.cvshome.org/docs/manual/cvs-1.11.16/cvs.html>

(substitute the current version for 1.11.16)